Universidad Zaragoza
1542

Trabajo Fin de Máster

Reducción de ruido en señales de video

Autor/es

Israel Aznar Marcos

Director/es

Adolfo Muñoz Orbañanos

Escuela de Ingeniería y Arquitectura (EINA)

2011

REDUCCION DE RUIDO EN SEÑALES DE VIDEO

RESUMEN

El trabajo realizado consiste en el diseño de un algoritmo de reducción de ruido teniendo como objetivo tres características principales: Que sea lo más genérico posible, requiera poca configuración manual y dar prioridad a la calidad del resultado obtenido, intentando en la medida de lo posible que sea un algoritmo rápido, siempre que ello no repercuta negativamente en la calidad de la imagen obtenida.

El algoritmo no está centrado en un solo tipo de ruido como es habitual, sino que permite la reducción de los tipos de ruido más comunes, centrándose en tres grupos:

- Ruido cromático: Grano y distorsiones en la información de color de la imagen.
- Ruido coherente: Es producido por señales de frecuencia determinada, que se han agregado a la señal de la imagen original produciendo distorsiones en la imagen.
- Ruido dinámico: Ruido cuyo contenido es variable en el tiempo, se da en señales de video.

Después de analizar los distintos tipos de ruido que suelen darse en una imagen, se han creado tres grupos de sub-algoritmos, cada uno especializado en uno de los tipos de ruido, para así obtener los mejores resultados posibles. Además, la solución propuesta permite ser usada tanto en fotografías como en señales de video.

El algoritmo se ha realizado de forma que sea lo más automático posible; realizando numerosas pruebas sobre casos prácticos, para determinar que valores de los parámetros de cada uno de los distintos sub-algoritmos, son los más adecuados para establecerse por defecto para que sean válidos para la mayoría de casos generales y reduciendo así el número de parámetros a configurar manualmente. Pero además también se permite un ajuste manual completo, para poder mejorar el resultado obtenido en los casos donde los valores por defecto no sean los más adecuados.

Para ser lo más genérico posible, algunas etapas del algoritmo permiten varios métodos de funcionamiento, estando cada uno especializado en variaciones de ruido concretas.

En la fase de implementación se ha tenido en cuenta la velocidad del algoritmo, probando diversos sub-algoritmos y librerías, y seleccionando lo que mejores resultados han dado y menos tiempo de procesamiento han tomado.

► Índice

► 1. Introducción

1.1 Descripción del problema

Las distorsiones no deseadas en fotografías y videos son un problema cotidiano, nuestras cámaras de fotos, cuando la iluminación es inferior a la adecuada, generan en la imagen miles de puntos con un tono de color y luminosidad que no es el adecuado, causando con ello una degradación en la calidad de la fotografía. Es algo similar a lo que en la fotografía tradicional se denomina grano y que en la fotografía digital es el ruido eléctrico.

También con la era digital, los contenidos en video se comprimen más, así se logra aprovechar todo el ancho de banda posible, para lo cual, se usan sistemas de alta compresión con pérdida de información. Estos producen degradación de la imagen, principalmente en la capa de color, produciendo unos artefactos llamados comúnmente macrobloques; esto no es exclusivo de los videos descargados de Internet, en las emisiones de televisión, sean terrestres o por satélite, también se produce aunque en menor medida, debido principalmente a su mayor ancho de banda y por lo tanto, menor ratio de compresión.

Además, cuando se trata la información mediante un proceso analógico, ésta es susceptible de que una o varias señales ajenas a la información original interfieran con esta y la alteren, produciendo ruido coherente. Podemos observar en la figura 1 un ejemplo de este tipo de ruido, el cual puede estar producido por una señal de una frecuencia determinada, o por la combinación de varias. Estas frecuencias además pueden variar con el tiempo. Este tipo de ruido es fácilmente reconocible en la mayoría de casos, pues producen en la imagen distorsiones en forma de un patrón repetitivo, el cual puede estar compuesto por secuencias de líneas, puntos, etc.

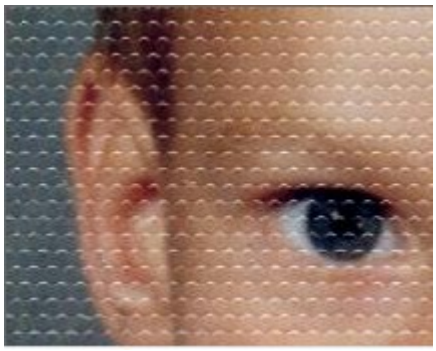

Fig. 1: Imagen con ruido coherente

1.2. Objetivo del trabajo

El objetivo que se pretende conseguir, consiste en un algoritmo general, el cual soporte la reducción de los principales tipos de ruido: grano en la información de color, ruido coherente y ruido dinámico. Permitiendo la reducción de este, tanto en imágenes estáticas como en señales de video.

Cada uno de estos tipos de ruido se debería procesar por separado, creando un sub-algoritmo específico para cada uno y adaptando si es preciso la información de la imagen, para así obtener los mejores resultados posibles.

El algoritmo debería estar preparado para procesar tanto imágenes estáticas, como una señal de

video, permitiendo en las señales de video adoptar algunas soluciones adicionales, como el análisis temporal.

El ruido dinámico se da en señales de video y consiste en píxeles de valor y posición aleatorios que cambian con el tiempo. Este tipo de ruido, es una parte importante donde debería centrarse el algoritmo y deberá ser lo más automático posible. Se debe evitar en lo posible la perdida de nitidez de la imagen y reducir las alteraciones no deseadas que se suelen producir con este tipo de algoritmos.

Otro punto importante es la reducción de ruido coherente (patrones de ruido), consistente en señales, normalmente fijas, que se han agregado a la imagen original. La detección de este tipo de ruido es compleja y su eliminación total no suele ser posible, dado que esta señal pasa a formar parte de la información de la imagen, por lo que en el proceso de reducción, hay también una perdida inevitable de información que no es ruido y desearíamos mantener. Se intentará dar con una forma de reducir en lo posible estos efectos secundarios y tener un equilibrio adecuado para todos los patrones de ruido.

1.3. Trabajos previos

Actualmente hay diversas soluciones que cubren las necesidades de reducción de los tipos de ruido descritos, las principales son:

− Kodak Digital GEM [8]: Especializado en la reducción de ruido cromático en fotografías; está disponible como un plugin de Photoshop y permite de forma sencilla, eliminar gran parte del ruido sin alterar en exceso el contenido real de la imagen.

− Neat Image [9]: Un software disponible tanto en aplicación independiente como en plugin de Photoshop; también tiene una versión del plugin especializada en video. Permite eliminar el ruido de tipo cromático y también el grano en la capa de luminosidad; es uno de los mejores algoritmos de reducción de ruido que hay en el campo fotográfico.

− Clear ID [10]: Un software especializado en análisis forense de fotografía y video, dispone de un buen sistema de reducción de patrones de ruido, uno de los mejores, pero completamente manual y se basa en el filtrado de un rango de frecuencias.

− Grid Pattern Removal (GPR) de Fujifilm [11]: Es un sistema con implementación por hardware y software, dedicado en exclusiva a la eliminación de patrones de ruido (ruido coherente), destinado al ámbito médico y especializado en radiografías; es un algoritmo que se basa en la eliminación de las muy altas frecuencias, por lo que solo es válido para casos específicos.

− Algoritmos de reducción de ruido 3D [12]: Son todos algoritmos muy similares, basados en un suavizado de la imagen tanto de forma espacial como temporal, son usados en sistemas de video, principalmente en los televisores modernos y sirve para eliminar el ruido dinámico que contienen las señales de video; se creó para reducir el ruido de las emisiones analógicas.

Estos son los principales sistemas empleados, cada uno especializado en un tipo de ruido. Todos tienen en común que están adaptados para una finalidad concreta.

Las soluciones destinadas al campo de la fotografía, todas ellas gestionan con eficacia el grano en la información de color e iluminación de la imagen. Sin embargo, no permiten eliminar el ruido coherente ni pueden ser aplicadas a señales de video para la eliminación de ruido dinámico.

Los algoritmos de reducción de ruido coherente, suelen ir destinados a casos muy específicos en los que se deben eliminar un conjunto de frecuencias concretas. Algunos permiten seleccionar manualmente el rango de frecuencias que se desea eliminar. El problema es que no están pensados para restaurar fotografías ni señales de video.

La solución propuesta, intenta proporcionar un algoritmo general para todos los tipos de ruido descritos, permitiendo ser usado tanto en fotografías como en señales de video.

Además, se intentará que el algoritmo sea lo más automático posible, estableciendo valores válidos para la mayoría de casos generales y reduciendo el número de parámetros a configurar.

► 2. Descripción general

2.1. Esquema general del algoritmo

El algoritmo se ha distribuido en varios grupos, en un esquema que podemos ver en la figura 2, cada uno especializado en la eliminación de un tipo de ruido y compuesto por uno o varios sub-algoritmos.

Puesto que algunos de estos sub-algoritmos están destinados al procesamiento de la información de color de la imagen y otros a la información de luminosidad, se precisará separar previamente la información de luminosidad y color de la imagen de entrada, para así poder ser procesadas de forma independiente.

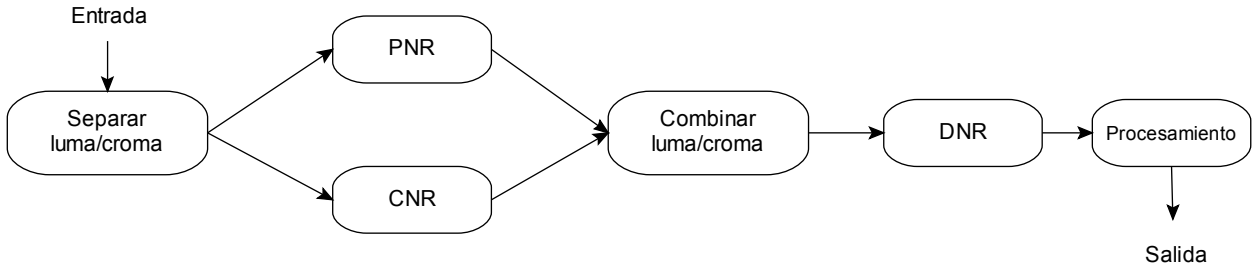

Fig. 2: Esquema general de las principales partes del algoritmo

Después de analizar los distintos tipos de ruido que suelen darse en una imagen, se han creado tres grupos de sub-algoritmos, cada uno especializado en un tipo de ruido, cada uno de estos grupos puede desactivarse, en cuyo caso la salida generada será el mismo contenido que se tiene a la entrada. A continuación se detallan los objetivos de cada uno de ellos:

Pattern Noise Reduction (PNR)

Este grupo es el responsable de la reducción de patrones de ruido (PNR), se encarga de reducir los niveles de ruido coherente, producidos por señales de una frecuencia determinada que se han agregado a la señal de la imagen original. Podemos ver un ejemplo de este tipo de ruido en la figura 3

Fig. 3: Muestra de ruido coherente

Chroma Noise Reduction (CNR)

El grupo de reducción de ruido cromático (CNR), se encarga de procesar la información de color de la imagen, para reducir el ruido y grano que esta contenga. En la figura 4 tenemos un ejemplo de ruido cromático.

Fig. 4: Muestra de ruido cromático

Dynamic Noise Reduction (DNR)

El grupo de reducción de ruido dinámico (DNR), está hecho específicamente para señales de video, no tiene efecto en fotografías. Se procesa tanto la componente de color, como la de luminosidad de la imagen, para reducir el nivel del ruido no coherente en la señal, sea de tipo aleatorio, ruido blanco o cualquier otro cuyo contenido sea variable en el tiempo. En la figura 5 podemos ver un ejemplo de este tipo de ruido.

Fig. 5: Muestra de ruido dinámico

Partiendo de la imagen de entrada, inicialmente ésta se separa en dos conjuntos de capas, uno con la información de luminosidad y otro con la de color. La información de luminosidad será procesada por el grupo PNR, mientras que la de color por el grupo CNR. Posteriormente, ambas ya procesaras, volverán a combinarse en una imagen completa para poder ser procesada por el grupo DNR. Finalmente, hay otro grupo de sub-algoritmos destinados a mejorar la apariencia visual del resultado final.

En la figura 6, podemos ver un diagrama más completo de la técnica propuesta, donde se muestran los distintos grupos y los sub-algoritmos de cada uno.

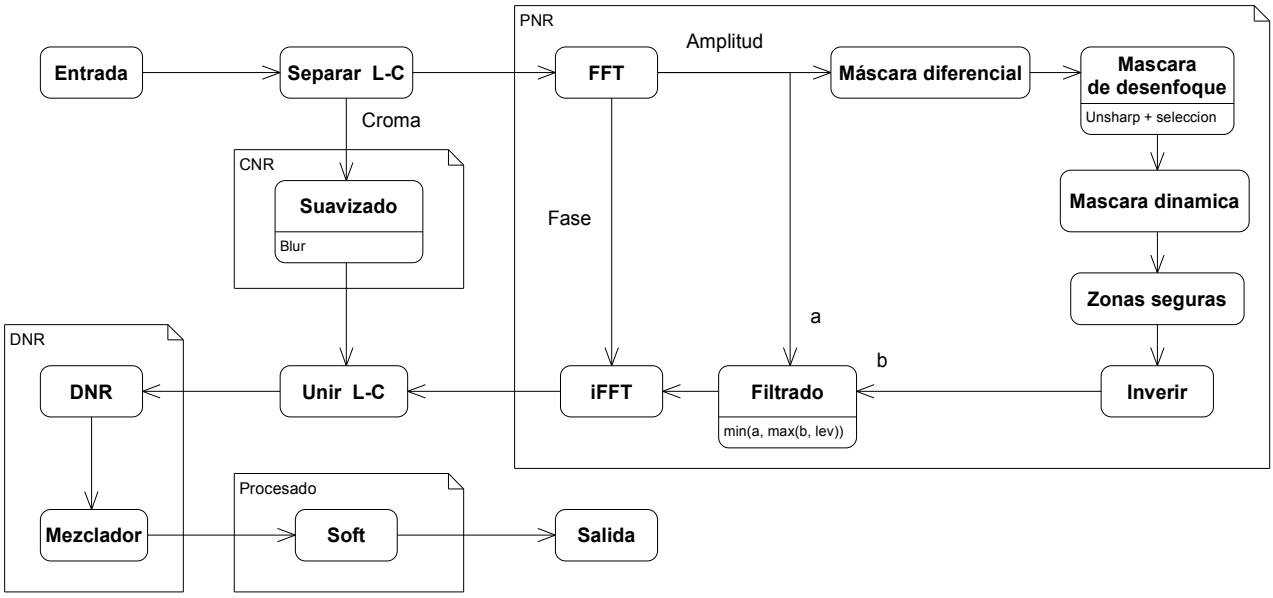

Fig. 6: Diagrama detallado de procesamiento

A continuación se detalla el funcionamiento y objetivos de cada uno de los principales grupos.

2.2. PNR: Reducción de ruido coherente

El grupo de sub-algoritmos de reducción de patrones de ruido (PNR), es el encargado de reducir los niveles de ruido coherente producidos por una o varias señales de frecuencia fija que se han agregado a la señal de la imagen original; para ello se procesa la información de luminosidad de la imagen, pues es donde se encuentra este tipo de ruido [13].

Consta de varias etapas, en la primera de ellas la información de la capa de luminosidad se convierte al espacio de frecuencia mediante la transformada de Fourier [1]. Esto se hace así porque en esta forma de representación de la información, podemos analizar y modificar las distintas señales que componen la imagen en base a su frecuencia, algo que para detectar y reducir el ruido coherente es indispensable.

Después de aplicar la transformada de Fourier, los datos son convertirlos en información que podamos manipular; esto consiste en convertir los datos complejos obtenidos de la transformada de Fourier, en dos capas de datos, una con la información de fase y otra con la información de amplitud. La capa de fase no la necesitamos modificar para nuestro propósito, por lo que no será alterada. La información que necesitamos modificar se encuentra en la capa de amplitud; es ésta la que contiene la información que nos es útil, pues nos indica la intensidad o nivel de cada una de las frecuencias, por lo que para reducir el nivel de una frecuencia en concreto, tan solo debemos modificar su valor en esta capa. Pero para ello, antes hay que averiguar que frecuencias son las que debemos modificar, detectando cuales no pertenecen a la imagen original y por lo tanto son de ruido agregado posteriormente, esto es posible gracias a que las señales que deseamos eliminar, salen reflejadas como zonas con una mayor intensidad que el valor medio de sus píxeles [2] adyacentes.

En la siguiente etapa, la capa de amplitud es analizada para crear una máscara; una máscara es un conjunto de valores, uno por cada píxel de la capa a la que hace referencia, la cual nos indica que conjunto de píxeles tenemos marcados o seleccionados para procesar o descartar y también nos puede indicar como debemos procesarlos o en que medida. En general, las máscaras aquí empleadas indican que contenido debe mantenerse intacto y cual debe modificarse.

Esta etapa consta de varios sub-algoritmos encargados de hacer la selección de que píxeles deben ser modificados, los primeros (máscara diferencial, máscara dinámica y resaltado mediante máscara de desenfoque), detectan que píxeles en la información de amplitud pertenecen a señales de ruido coherente y por lo tanto deben ser procesados. El último, llamado zonas seguras, es el cargado de proteger algunas zonas que contienen información delicada y que no debe ser modificada o causaría grandes alteraciones no deseadas en la imagen.
A continuación se resume el objetivo y funcionamiento de cada uno de estos sub-algoritmos:

– Máscara diferencial: Crea la máscara partiendo de los datos complejos proporcionados por la transformada de Fourier. Se usa una fórmula para la conversión de los datos. Es un método completamente automático que no precisa parámetros de configuración.

– Máscara dinámica: Es un sub-algoritmo de corte, filtra los píxeles que no tenga un nivel mínimo de intensidad. Además tiene un modo de detección temporal (por lo tanto solo valido para señales de video), en el que se analiza el contenido de la imagen actual y lo compara con el de la anterior, para así determinar que partes de la imagen son comunes y crear la máscara en base a ello.

– Resaltado mediante máscara de desenfoque: Mediante el sub-algoritmo de máscara de desenfoque [3] aplicado a la capa de luminosidad, obtenemos una máscara donde se indica el

contenido a mantener y también se realza el contraste de las zonas a ser modificadas.

– Zonas seguras: En la capa de magnitud hay contenido que no debe ser modificado, por ello se modifica la máscara para asegurarse que no sea alterado.

Una vez que ya tenemos generada la máscara mediante los sub-algoritmos anteriores, se prepara la información para la fase de filtrado, para ello, se invierte el valor de los píxeles de la máscara y se modifica su valor en el sub-algoritmo de procesamiento, en el cual cada uno de los valores de amplitud se intensifica o se deja intacto según se indique en la máscara.

La capa de amplitud modificada y la máscara se pasan a la fase de filtrado junto a la capa de amplitud original; en esta fase se compara la información de ambas capas y se selecciona solo la de menor amplitud.

La información de salida de la fase de filtrado y la capa de fase, se procesan mediante la transformada inversa de Fourier [1] para obtener una nueva capa de luminosidad, en la cual el ruido coherente habrá sido reducido.

2.3. CNR: Reducción de ruido cromático

El sub-algoritmo de reducción de ruido cromático (CNR), es el encargado de reducir el ruido y grano que contenga la información de color de la imagen.

Dado que el ojo humano percibe peor los cambios de color que las variaciones de luminosidad, se puede reducir en gran medida la información de color sin que el ojo perciba una pérdida de calidad.

Este sub-algoritmo realiza una mezcla de los píxeles de las capas de color, con esto se consigue eliminar en gran medida el ruido cromático que éstas puedan tener.

No es habitual que una señal de ruido afecte solo a la capa de color, normalmente también pueden verse los efectos del ruido en la capa de iluminación, por lo que generalmente también se deberá tratar ésta mediante un algoritmo especializado.

2.4. DNR: Reducción de ruido dinámico

El algoritmo de reducción de ruido dinámico (DNR), se encarga de reducir el tipo de ruido no coherente de la imagen, que es común a las capas de color e iluminación. Es de tipo mezclador temporal, razón por la cual solo tiene efecto en señales de video y no en imágenes estáticas.

Partiendo de la información de la imagen actual y la de las imágenes previas, éstas se comparan y se selecciona el contenido similar; se determina que es similar basándose en la diferencia existente entre los valores de cada uno de los píxeles del contenido actual y el previo, entonces los píxeles de cada capa correspondientes a la imagen actual y anterior se mezclan. Esto hace que los cambios de valor entre los píxeles de la imagen actual a la siguiente se vean atenuados.

► 3. Detalle del algoritmo

3.1. Separación de los canales de luminosidad y color

El primer paso antes de procesar cada tipo de ruido, es adaptar los datos de entrada a las necesidades de cada sub-algoritmo, para ello, lo primero es separar la información de la imagen en sus componentes de iluminación y color, ya que las vamos a procesar por separado con un sub-algoritmo especializado en un tipo de ruido distinto.

Un modelo de color, es una forma de representar los colores que puede tener un píxel. El modelo de color de origen es RGB [5], en el cual cada píxel es representado por tres valores llamados componentes; uno contiene el nivel de color rojo, otro el de azul y el otro el de verde.

Pero este modelo de color, no es el más adecuado para algunos sub-algoritmos de procesamiento empleados, pues la información de color e iluminación no están separadas, por lo que si queremos procesarlas de forma individual deberemos convertir a otro modelo de color.

Se han probado varios modelos de color para separar la información de iluminación y color, como el HSL [5], pero no se han obtenido buenos resultados, los colores se alteraban en exceso y había perdida de información; el modelo de color que mejor ha funcionado en las pruebas ha sido el YUV [5], siendo la Y la componente de iluminación y los canales U y V contienen la información de color.

El canal de iluminación Y viene dado por:

$$Y = R * 0.299 + G * 0.587 + B * 0.114$$

Los canales de color U y V vienen dados por:

$$U = R * -0.168736 + G * -0.331264 + B * 0.5 + 128$$

$$V = R * 0.5 + G * -0.418688 + B * -0.081312 + 128$$

En las figuras 8, 9 y 10 podemos ver los canales Y, U y V respectivamente, que obtenemos de la imagen original (figura 7). Se ha ajustado el nivel de brillo en las figuras 9 y 10 para una mejor apreciación, ya que originalmente son muy oscuras.

Fig. 7: Imagen original, con todos los canales combinados

Fig. 8: Canal Y, que contiene la información de luminosidad

Fig. 9: Canal U de color

Fig. 10: Canal V de color

3.2. PNR: Reducción de ruido coherente

Ahora que tenemos separada la información de la imagen en sus componentes de color y luminosidad, nos centramos en la segunda, que es la capa sobre la que se va a aplicar el algoritmo de reducción de ruido coherente.

3.2.1. Descomposición de la capa de luminosidad

Para poder procesar la componente de luminosidad (figura 11), es preciso convertir la información de esta capa al espacio de frecuencia, para así poder reducir o eliminar el contenido de algunas frecuencias concretas; esto es muy útil pues el ruido coherente se puede tratar como una distorsión de una o varias frecuencias y de esta forma podremos detectarlas.

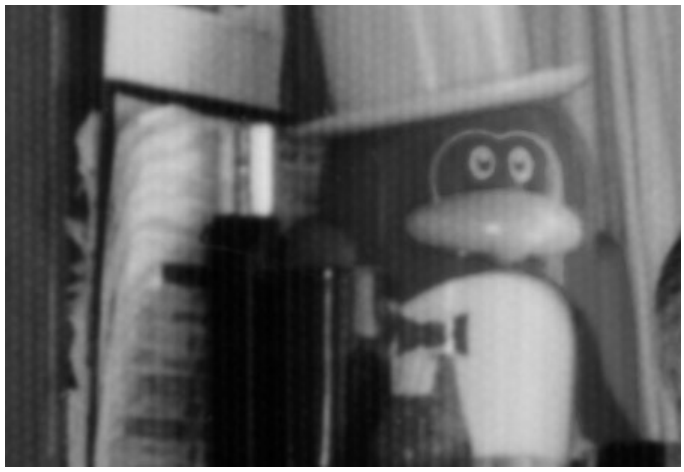

Fig. 11: Capa de luminosidad

Para realizar esta conversión usaremos la transformada discreta de Fourier (DFT) [1], concretamente el algoritmo de la transformada rápida de Fourier (FFT), la cual, a partir de una capa con la información de luminosidad de la imagen, nos proporciona otra con valores complejos representando coordenadas cartesianas [6] en el espacio de frecuencia.

Esta capa obtenida debemos transformarla para poder trabajar con ella, debemos convertir los datos en coordenadas polares [6]; se convierte cada valor complejo en dos valores, uno con el valor de fase y con otro con el de amplitud; crearemos dos nuevas capas con estos valores por separado, teniendo así una capa de fase y otra de amplitud (figura 12).

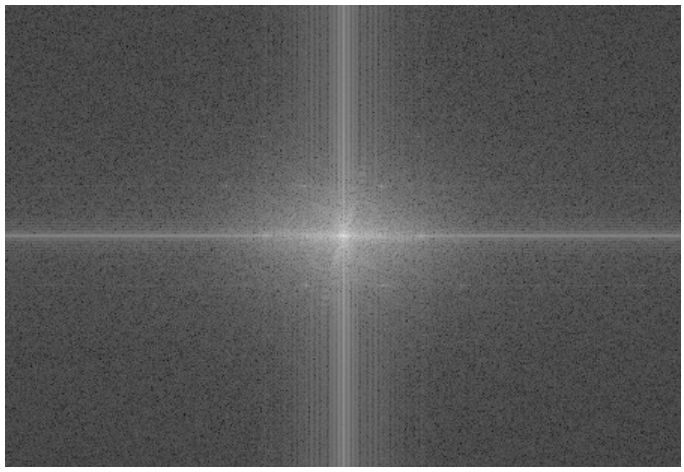

Fig. 12: Canal de amplitud de las frecuencias que componen la capa de luminosidad

Estas conversiones vienen dadas por:

$$amp = \frac{n * \log(|val| + 1)}{dma} \quad \text{con} \quad dma = \log(isize * n + 1)$$

$$fase = n * \arctan(\Im / \Re)$$

Siendo *val* el valor imaginario a convertir, *isize* la cantidad de valores en la capa, que en nuestro caso es el número de píxeles de la imagen y *n* el valor máximo que puede tener un píxel de la capa de amplitud; si *n* vale 1 se obtienen valores normalizados.

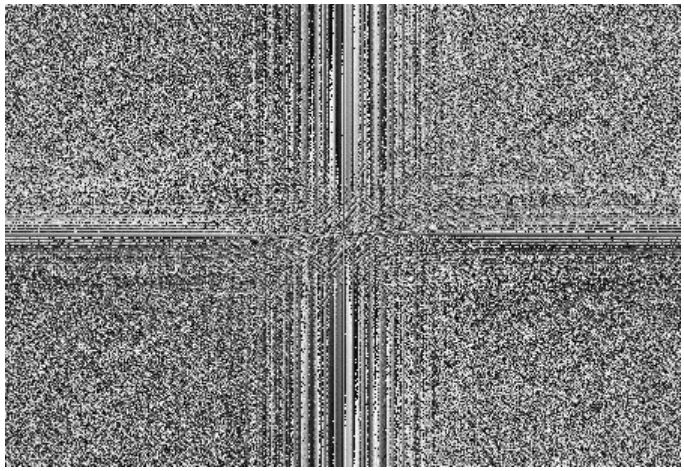

Fig. 13: Canal de fase de las frecuencias que componen la capa de luminosidad

En la figura 11 se puede ver la capa de luminosidad que tenemos a la entrada del sub-algoritmo y también tenemos las dos capas de salida, la de amplitud (figura 12) y la de fase (figura 13); para nuestro sub-algoritmo de reducción de ruido coherente solo necesitamos modificar la capa de amplitud.

Una vez que tenemos las capas de amplitud y fase, el siguiente paso consiste en crear una máscara que indique que píxeles de la capa de amplitud deben ser modificados y de que forma. Para ello se pueden emplear varios sub-algoritmos, cada uno especializado en un método de selección:

Máscara diferencial, máscara dinámica y resaltado mediante máscara de desenfoque.

Estos sub-algoritmos son de uso opcional, pues se puede no usar ninguno, además están ideados para que solo sea utilizado uno de ellos, pero el algoritmo permite el uso de varios simultáneamente, donde cada uno toma como capa de entrada la máscara generada por el anterior.

Posteriormente hay otro sub-algoritmo llamado zonas seguras, que se encarga de proteger ciertas zonas de la capa de amplitud donde hay contenido que no debe ser modificado, para ello modifica la máscara impidiendo la selección de píxeles dentro de esas zonas.

La máscara objetivo a generar a partir de la capa de iluminación de la figura 12, es la que podemos ver en la figura 14; en ella podemos ver los datos que realmente nos interesan; son las pequeñas zonas de alta amplitud generadas por el ruido coherente y que podemos ver representadas por zonas de alta luminosidad.

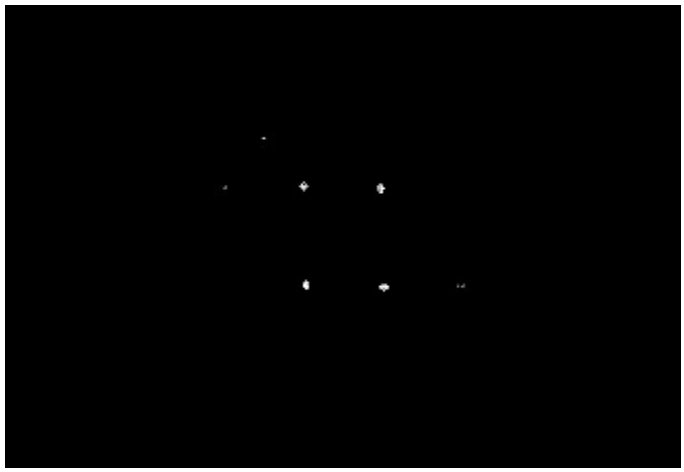

Fig. 14: Datos que realmente nos interesan. Esta es la máscara objetivo (generada manualmente)

En la figura 15 tenemos una imagen con solo la señal de ruido coherente, imagen que obtendríamos dejando como capa de amplitud la imagen de la figura 14. Esta es la señal que deseamos eliminar de la imagen, por lo que debemos conseguir reducir esta parte de la información afectando al resto lo menos posible.

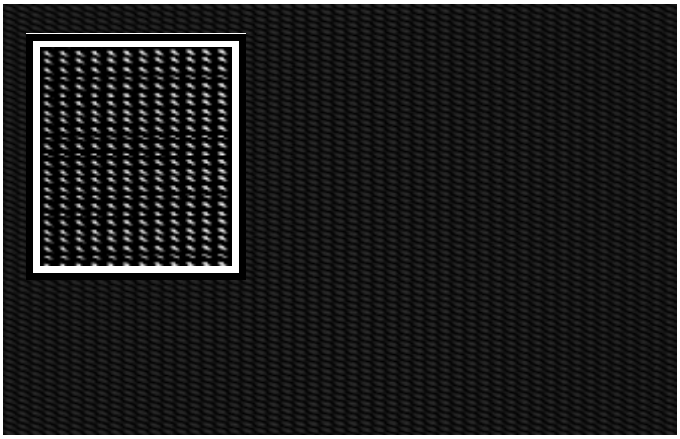

Fig. 15: Imagen con solo el ruido coherente. Arriba - izquierda: Ampliación y resaltado de una parte para una mejor apreciación

A continuación pasamos a ver los tres sub-algoritmos encargados de generar la máscara de selección.

3.2.2. Máscara diferencial

El sub-algoritmo de máscara diferencial crea la máscara partiendo de los datos complejos proporcionados por la FFT, del mismo modo que hemos generado la capa de amplitud creamos la máscara de selección, solo que el valor de cada píxel de la máscara viene determinado por:

$$v = \frac{n * |val|}{isize}$$

Donde *val* es el valor complejo a convertir, *n* el máximo valor que puede tener un píxel de la máscara e *isize* el número de píxeles en la capa de datos.

Este sub-algoritmo no precisa configurar ningún parámetro por lo que tiene la ventaja de ser completamente automático.

En la figura 16, podemos ver la máscara generada mediante el sub-algoritmo de máscara diferencial a partir de la capa de luminancia de la figura 11. Como podemos apreciar al resultado es muy diferente de la capa de amplitud calculada anteriormente (figura 12), todos los píxeles con un valor medio pasan a tomar un valor próximo a 0 (representados por tonos oscuros) y en la máscara solo se han seleccionado las zonas de mayor intensidad.

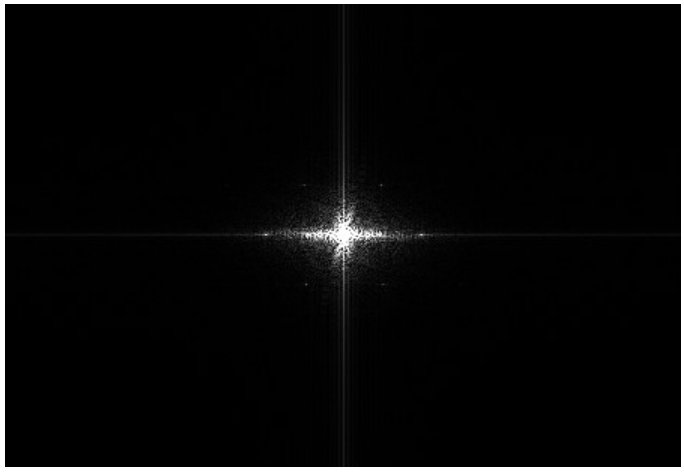

Fig. 16: Máscara generada mediante el sub-algoritmo de máscara *diferencial*

Este sistema tiene la ventaja de ser automático y dar buenos resultados con señales de ruido coherente intensas, las cuales salen representadas en la capa de amplitud por zonas de alta intensidad, además permite detectar tanto picos de alta intensidad en una o múltiples frecuencias como zonas más amplias que representan un rango de frecuencias mayor.

Sus limitaciones vienen dadas porque solo detecta las partes más intensas de la señal del ruido coherente, que aunque en la mayoría de ocasiones es suficiente para obtener una máscara que permite obtener un resultado equilibrado, donde se reduzca la mayoría del ruido sin producir casi perdida de nitidez en la imagen; también hay casos donde el ruido eliminado no es suficiente, pues hay píxeles que contienen información de la señal de ruido que no se han seleccionado debido a una baja intensidad.

3.2.3. Máscara dinámica

La máscara dinámica es otro de los sub-algoritmos destinados a seleccionar qué datos del canal de amplitud son los que debemos modificar y también en qué medida.

Tiene dos modos de funcionamiento simultáneos, uno espacial y otro temporal que solo tiene utilidad en señales de video.

– El modo de procesamiento espacial, siempre está habilitado y es en el que se analiza la información; primero se hace una selección de píxeles en base a un valor mínimo *minl*, los píxeles cuyo valor estén por debajo de *minl* serán ignorados y por lo tanto no serán alterados; esto se hace estableciendo en la máscara estos píxeles a un valor próximo al 0, equivalente en la imagen a un tono de color negro.

– El modo temporal: solo tiene efecto en señales de video, la selección de los datos a procesar se realiza mediante un sistema diferencial; por cada píxel, se compara la diferencia de los píxel de esa misma posición entre dos cuadros consecutivos y si esta es superior a una constante *difl*, entonces se selecciona para ser procesado.

El fundamento de estos métodos de selección de datos, se basa en que el ruido coherente viene representado en la capa de amplitud (figura 17), como zonas donde los píxeles tienen un valor superior al de la media de sus píxeles adyacentes, por lo que son zonas contrastadas y además, como la señal de ruido coherente está constituida por una o varias frecuencias fijas, la posición de esas zonas será la misma en dos cuadros consecutivos; por lo que determinamos que solo los píxeles que cumplan con la primera o ambas de estas dos condiciones pertenecen a una señal de ruido coherente y entonces deben ser procesados para reducirlo.

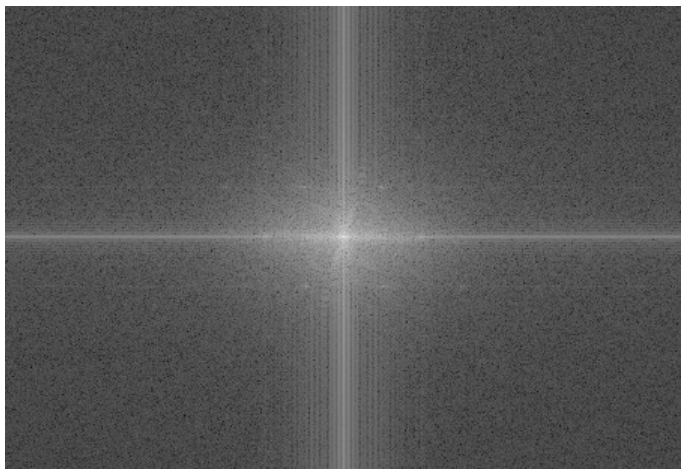

Fig. 17: Capa de amplitud

El valor de los píxeles seleccionados en la máscara serán modificados; se puede establecer el nuevo valor con una constante fija: $nuevo = cnt$ con $0 \le cnt \le 255$

O bien se establece una constante que haga de multiplicador por el valor actual: $nuevo = min((actual * cnt) \div 50, 255)$ con $0 \le cnt \le 255$

En la mayoría de casos prácticos ambos modos de modificación dan resultados similares, pero el usar una constante en lugar de un multiplicador suele producir una mayor distorsión no deseada en el contenido real de la imagen.

En la imagen que representa la capa de amplitud, contra mayor sea el nivel de cada píxel, este saldrá representado por un tono más claro, llegando hasta el blanco en su valor máximo.

En la figura 18, podemos ver los datos seleccionados usando una constante de valor 110 para el parámetro *minl*, tomando como origen la capa de amplitud de la figura 17, perteneciente a una imagen estática. Pero todavía hay que hacer una selección más precisa, por lo que aumentamos el valor de la constante hasta 141, con ello llegamos a tener una selección como la de la figura 19.

Posteriormente aumentamos la constante de multiplicación, para incrementar el nivel de los datos seleccionados y generar así los nuevos valores de la máscara que podemos ver en la figura 20.

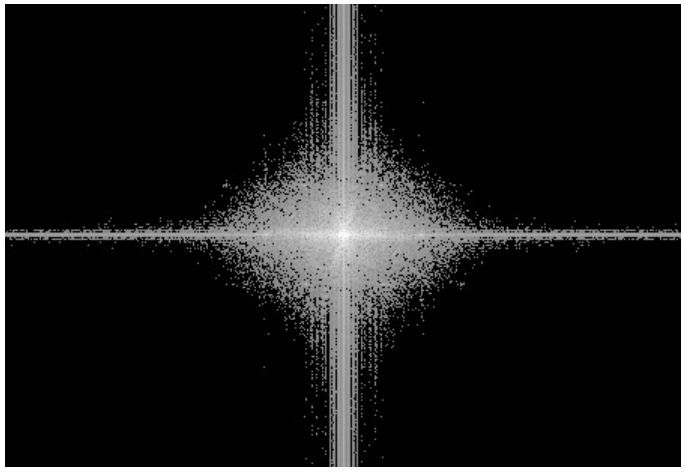

*Fig. 18: Máscara con los datos seleccionados en base a un nivel mínimo de 110 (*con *minl en rango de 0 a 255)*

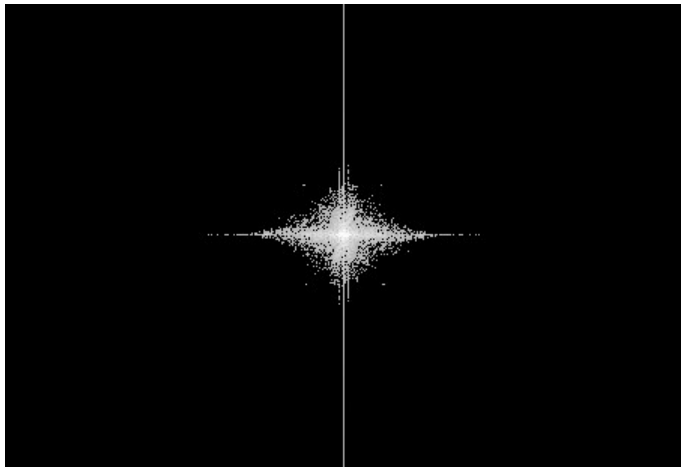

Fig. 19: Máscara aumentando el valor mínimo (minl) a *141*

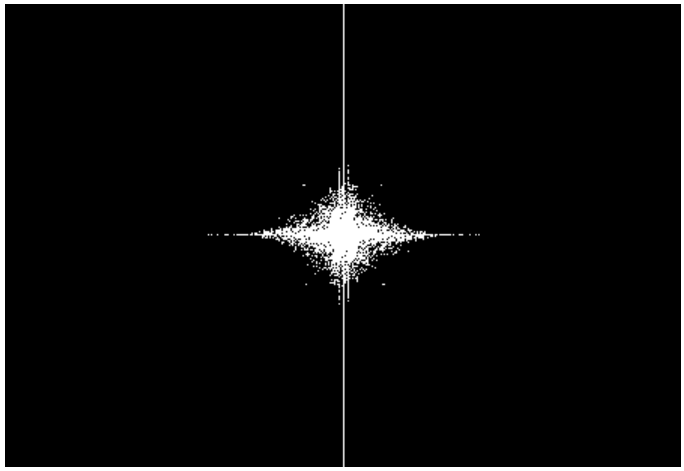

Fig. 20: Máscara usando un multiplicador cnt = 68 (factor = 1,36)

3.2.4. Resaltado mediante máscara de desenfoque

La máscara de desenfoque [3], es un algoritmo para acentuar los contornos y resaltar zonas de una imagen. Su objetivo es aumentar la nitidez aparente de una imagen, para lo cual se aumenta la acutancia, es decir, se aumenta la diferencia de luminosidades en la transición de tonos de la imagen.

Ya que el ruido coherente sale representado como grupos de píxeles con un valor superior a la de los píxeles adyacentes, los cuales representan a un conjunto de frecuencias, este algoritmo nos es especialmente útil para detectar esas zonas y resaltarlas.

Partiendo de la capa que representa los datos de amplitud en el espacio de frecuencia (figura 21), queremos resaltar las zonas de mayor amplitud; éstas están representadas por los tonos más cercanos al blanco, por el contrario contra más oscuro sea el tono, menor amplitud.

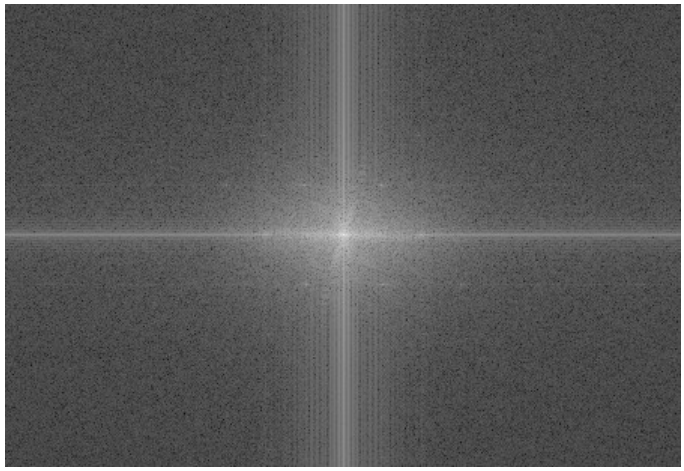

Fig. 21: Imagen original, antes de ser procesada

En el algoritmo de máscara de desenfoque hay varios parámetros de entrada:

– La intensidad *int* con la que aplicar el efecto de resaltado.

– El radio *rd* mínimo de las zonas (indicado en píxeles) para que el algoritmo las resalte.

– El valor *dif* mínimo de la diferencia de luminosidad entre dos píxeles vecinos, a partir del cual será aplicado el efecto.

Una máscara de desenfoque consiste en duplicar el contenido de la capa, a esta copia realizar un suavizado Gaussiano [4], y después restarla a la capa original.

Si aplicamos una máscara de desenfoque [3] con un valor de intensidad *int* = 100, un radio pequeño *rd* = 3 y un umbral mínimo *dif* = 0, obtenemos que todas las zonas de la imagen son resaltadas, tal y como podemos observar en la figura 22.

Fig. 22: Máscara generada usando un radio pequeño, rd = 3; podemos ver como se han resaltado todos los píxeles de forma independiente

Aunque en la imagen resultante, tenemos resaltadas todas las zonas de mayor magnitud, no es el resultado que nos interesa, pues necesitamos detectar solo zonas de mayor anchura y no todas las

pequeñas variaciones de amplitud.

Así pues, probamos a aumentar el valor del parámetro de radio a *rd* = 64, para que el filtro se centre en estas zonas.

Al aumentar el radio, ahora sí tenemos más resaltadas las zonas de mayor anchura tal y como podemos ver en la figura 23, pero todavía debemos evitar el resaltado en las zonas de menor amplitud, ya que estos datos no queremos que sean modificados.

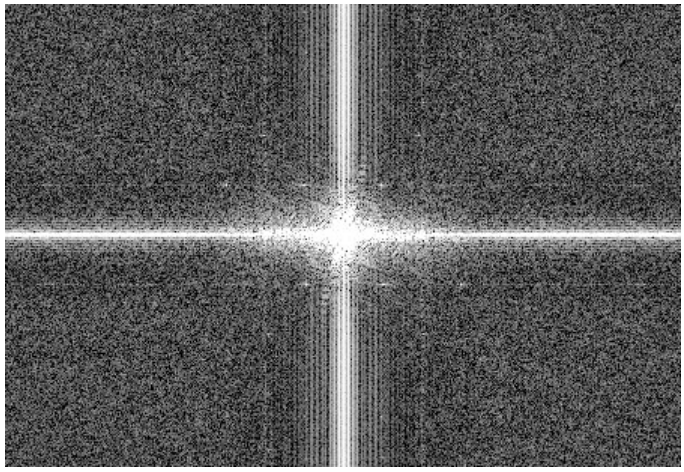

Fig. 23: Máscara generada con un radio mayor, rd = 64

Para evitarlo aumentamos el valor umbral mínimo a *dif* = 23, que controla la diferencia mínima de luminosidad con la que será aplicado.

Como podemos ver en la figura 24, ahora sí que tenemos resaltadas las zonas que nos interesan.

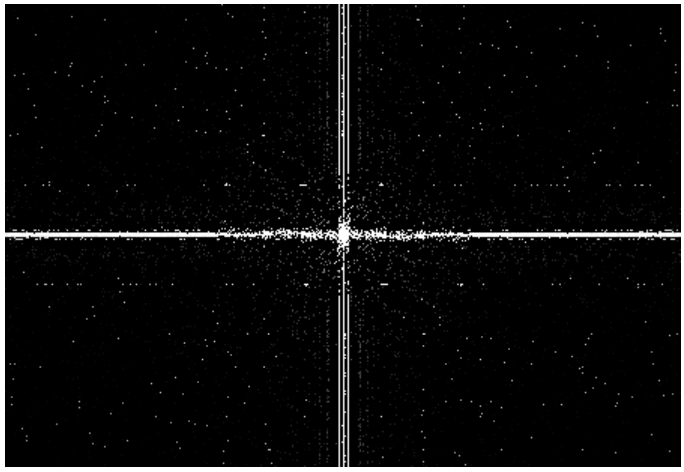

Fig. 24: Máscara generada aumentando el umbral mínimo, dif = 23

El algoritmo de máscara de desenfoque se ha modificado para que todos los píxeles no alterados sean filtrados, para ello se establece su valor al mínimo, representado en la imagen por el color negro.

De los sub-algoritmos destinados a la creación de la máscara, este es el que en la mayoría de casos consigue seleccionar una mayor cantidad de píxeles pertenecientes a la señal de ruido coherente, pero también selecciona muchos que no pertenecen a ella, por lo que elimina información real de la imagen; además es el sub-algoritmo más lento y el que mayor ajuste manual requiere.

3.2.5. Zonas seguras

En la máscara de selección de los píxeles de amplitud obtenida mediante uno o varios de los algoritmos anteriores, además de las zonas con ruido, representadas por las pequeñas zonas de mayor luminosidad, hay zonas especialmente importantes, pues seleccionan partes que contienen gran cantidad de la información de la imagen que no deseamos eliminar, una de estas zonas es el centro del eje de coordenadas, en este caso está en el centro de la imagen y suele ser la zona más luminosa. Para este ejemplo partimos del resultado del algoritmo anterior (figura 24)

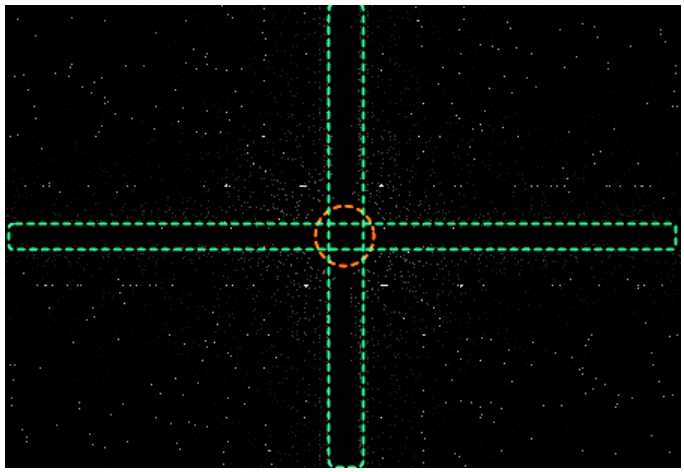

Fig. 25: Zonas seguras que normalmente deben ser protegidas

Para evitar la modificación no deseada de estas zonas, está el algoritmo de zonas seguras, que se encarga de proteger estas partes de la máscara. Permite gestionar las tres principales zonas importantes de la representación de amplitudes; el centro del eje de coordenadas, la vertical sobre el centro y la horizontal sobre el centro; representadas en la imagen por las zonas en forma de cruz.

Cada una de las tres zonas seguras que gestiona este algoritmo (figura 25), pueden ser controladas de forma independiente, tanto el grosor de éstas, como si deseamos permitir o no su procesado. El grosor está indicado en tanto por mil respecto a las medidas de la imagen; un valor de 500 en las secciones vertical y horizontal (conjunto denominado *cruz*) hará que tomen como grosor la mitad de la altura y anchura de la imagen respectivamente; si optamos por no mantener la relación de aspecto, se tomara en ambos casos el mismo valor, estando relacionado este con el mayor valor entre la anchura y altura de la imagen.

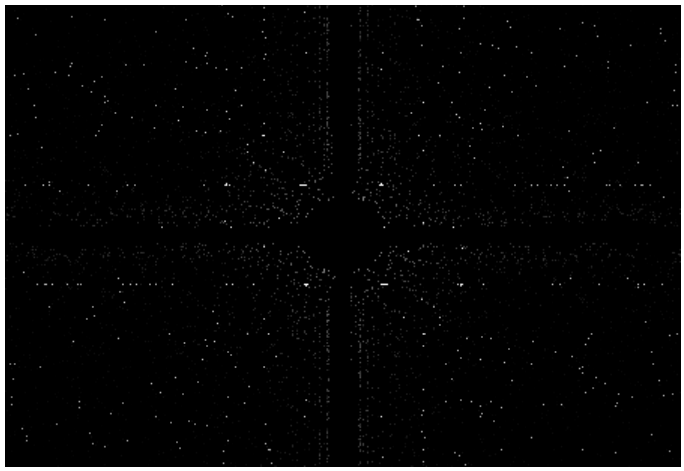

Fig. 26: Aplicando las zonas seguras con unos valores de: centro = 80, cruz = 58, manteniendo relación de aspecto.
Podemos ver que ahora solo tenemos resaltados los puntos que queremos filtrar.

El tamaño de la zona central, viene indicado de la misma manera respecto al menor valor entre el ancho y alto de la imagen.

Aunque por norma general, se ha visto que lo más indicado es proteger las 3 zonas, hay casos en los que hay que dejar que se procese alguna de ellas para obtener una mejor reducción del ruido.

En la figura 25 podemos ver resaltadas las distintas zonas seguras, la parte central y más importante es la de color naranja.

Como podemos ver en la figura 26, después de aplicar el algoritmo de zonas seguras, la imagen solo contiene las zonas que deseamos procesar, representadas en tonos claros.

3.2.6. Inversión de la máscara

La finalidad de este sub-algoritmo, es invertir los píxeles de la máscara, para adaptarlos al posterior filtrado de la etapa siguiente.

Para cada píxel se hace: $d = max(type(d)) - d * gint$, donde *d* es el valor de cada píxel de la capa y *type(d),* representa el tipo usado para almacenar el valor de la información de un píxel, por lo que *max,* representa el valor máximo que puede tener el tipo usado para representar el valor y *gint* es una constante que indicar el nivel de reducción de ruido.

Así pues, en la figura 27, podemos ver una representación de la inversión de los datos que teníamos en la figura 26.

En esta representación, los tonos oscuros son los valores que queremos reducir o eliminar, mientras que contra más claro sea el tono, menor será el cambio que se produzca en él.

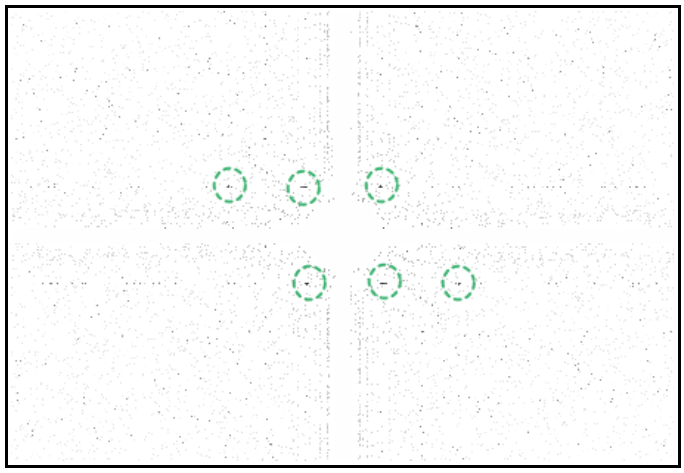

Fig. 27: Datos invertidos. Las pequeñas zonas oscuras remarcadas con círculos verdes son los datos que serán filtrados y pertenecen a la señal de ruido coherente.

3.2.7. Selección mediante máscara (filtrado)

La selección de los píxeles para la imagen final, se obtiene a partir de la capa de amplitud original y la obtenida del algoritmo de inversión; se seleccionan los valores de amplitud más bajos entre ambas capas y se genera con ellos una nueva capa de amplitud.

Para evitar la eliminación o filtrado completo de ciertas frecuencias, ya que estas no solo contienen información del ruido, sino también de los datos válidos, podemos establecer un valor mínimo de amplitud para los nuevos datos, al que denominaremos *lev*, este valor puede ser una constante general o bien un porcentaje del valor original.

El filtrado viene dado por: $min(a, max(b, lev))$

Donde *a* es el conjunto de valores de la capa original, *b* el conjunto de valores de los datos procesados e invertidos y *lev* el valor mínimo de filtrado. En este caso se ha usado *lev* = 0.

El resultado de esto será una capa de amplitudes, en la que los datos que hemos seleccionado en las etapas previas son modificados en la capa original, reduciendo así la amplitud de las frecuencias que producen el ruido.

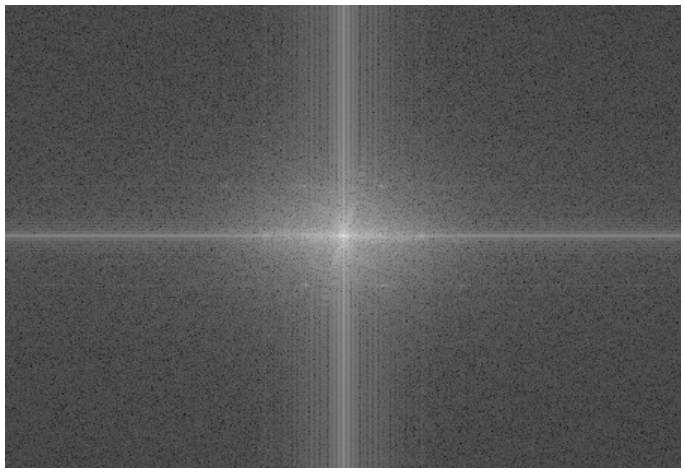

Fig. 28: Datos originales, destacar las pequeñas zonas de mayor luminosidad que queremos filtrar

En la figura 28 podemos ver la capa de amplitud original, antes de ser procesada y en la figura 29 tenemos la misma capa después del proceso de filtrado.

Podemos ver que los picos de amplitud de frecuencia que teníamos anteriormente han sido eliminados.

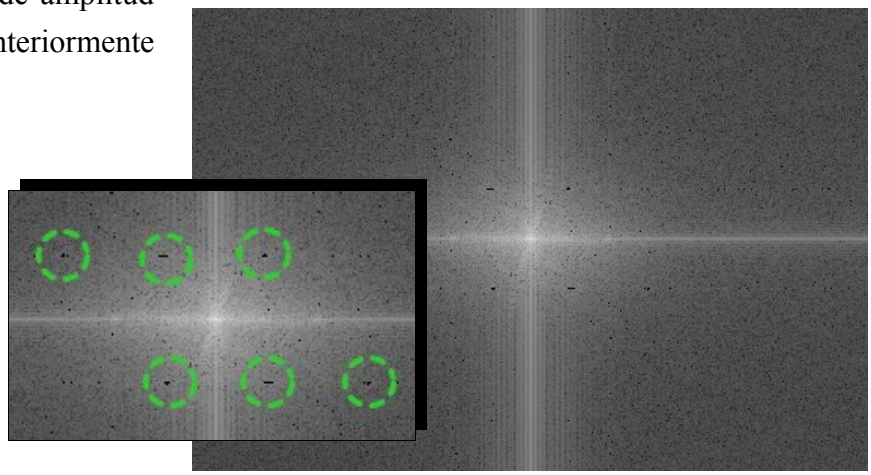

Fig. 29: Datos filtrados, las pequeñas zonas oscuras resaltadas con círculos verdes son los datos que se han filtrado

3.2.8. Composición de la nueva capa de luminosidad

A partir de los nuevos datos de amplitud obtenidos en las fases anteriores, donde hemos filtrado las frecuencias no deseadas, ahora debemos generar de nuevo un canal de luminosidad, para lo cual haremos la transformada inversa de Fourier con la capa de amplitud modifica y la capa de fase original, para hacer el proceso de conversión inverso y así volver a la imagen en espacio normal.

En la figura 30 podemos ver la capa original de luminosidad, donde se observa el ruido coherente en la imagen, que en este caso genera líneas verticales a lo largo de toda la imagen.

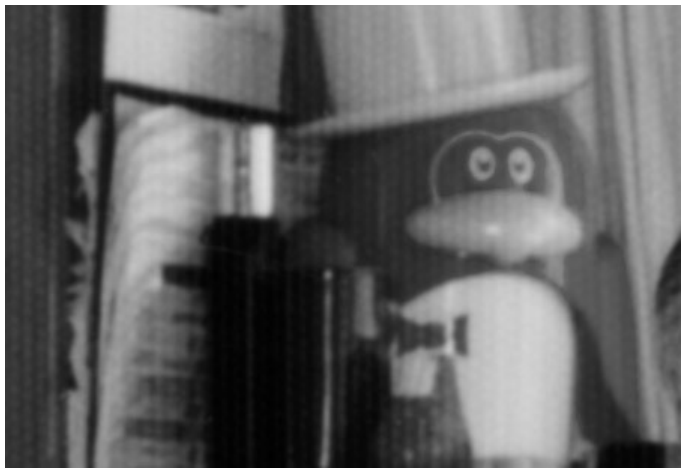

Fig. 30: Capa de luminosidad original

Como se puede observar en la figura 31, la nueva capa de luminosidad que hemos generado, muestra que la intensidad de las líneas verticales se ha reducido.

Si aumentamos el nivel de filtrado aplicado, no solo reduciremos el nivel de ruido, también eliminaremos información real de la imagen que no pertenece a la señal de ruido y por lo tanto habrá una perdida en el nivel de detalle y se podrían producir distorsiones no deseadas en la imagen. Por ello es importante tener un equilibrio en el nivel de filtrado, para así obtener una reducción de ruido razonable, manteniendo el nivel de detalle de la imagen.

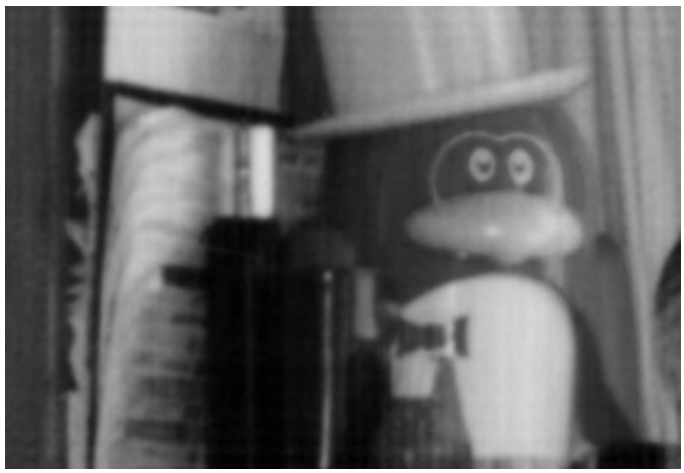

Fig. 31: Capa de luminosidad procesada, se puede apreciar que el ruido en forma de líneas verticales se ha visto reducido

3.3. CNR: Reducción de ruido en las capas de color

En este sub-algoritmo se hará una reducción del ruido en la capa de color. La información de color es mucho más sencilla de tratar que la de iluminación, pues el ojo humano tiene menor sensibilidad a esta, por lo que se pueden hacer grandes cambios sin que apenas se note; de hecho, en base a esta característica del ojo, se han realizado números algoritmos de compresión basados en eliminar parte de la información de color en imágenes y videos, como el JPEG [14] y el MPEG [7,15].

El tipo de ruido que deseamos reducir en las capas de color, son principalmente píxeles con un tono de color muy distinto a los de su alrededor y que por lo tanto tienen un tono contrastado, para reducirlo, se aplicara un efecto de suavizado sobre cada uno de los canales de color de forma independiente.

El efecto de suavizado, consiste en que el nuevo valor de cada píxel depende no solo de su valor actual, sino también del de los píxeles de su alrededor, la proporción de esta mezcla puede venir definida por una función o por una matriz, depende del filtro que se use.

En este caso se usa un suavizado Gaussiano [4], principalmente por su velocidad.

El parámetro que determina la intensidad con la que el efecto es aplicado en este tipo de algoritmo, es el radio, que indicara la distancia máxima de los píxeles vecinos con la que se realizara la mezcla de cada píxel.

El radio establecido es importante, pues un radio muy pequeño apenas eliminara el ruido y un valor elevado generara halos y defectos en el color de la imagen. Un radio de 2 a 3 píxeles es el más indicado en la mayoría de los casos.

En la figura 32 tenemos la imagen original del canal de color V y en la figura 33, podemos ver este mismo canal una vez procesado con un radio adecuado.

Fig. 32: Canal de color V antes de ser procesado

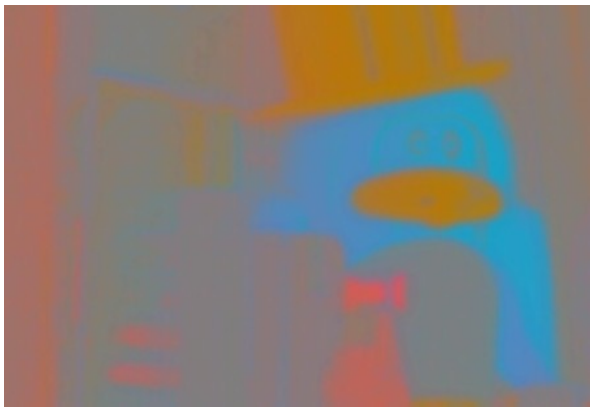

Fig. 33: Canal de color V después del procesado con un radio adecuado (2 píxeles)

Por el contrario, en la figura 34 podemos ver el resultado obtenido por un valor de radio demasiado alto en uno de los canales de color, este sobre-suavizado tendrá como consecuencia la distorsión de color en la imagen final.

Podemos comprobar este efecto en la figura 35, donde se ve un suavizado excesivo a un canal de color.

Por ello hay que mantener un equilibrio entre la cantidad de ruido cromático a eliminar y la perdida en la información de color que produce el sub-algoritmo.

Fig. 34: Imagen resultante con un radio elevado en el canal V (arriba) y U (abajo); radio = 12

Fig. 35: Canal de color U después del procesado un radio elevado; radio = 12

3.4. Combinación de los canales de luminosidad y color

Una vez que ya tenemos las capas de luminosidad y de color procesadas de forma independiente, volvemos a combinarlas para obtener una imagen completa, para seguidamente poder procesar el ruido común de las dos, como por ejemplo el ruido dinámico.

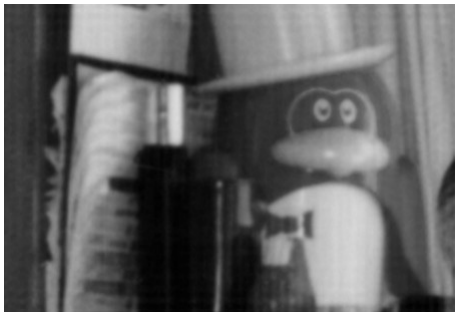

Fig. 36: Canal de luminosidad

Tenemos 3 canales de origen en el espacio de color YUV: el canal de luminosidad (figura 36), el canal de color V (figura 37) y el canal de color U (figura 38). Debemos combinarlos en una imagen en el espacio de color RGB, que es el que necesitamos para la siguiente etapa de procesado.

Fig. 37: Canal de información de color V

Las fórmulas de conversión de YUV a RGB son:

$$B = 1.164(Y-16) + 2.018(U-128)$$

$$G = 1.164(Y-16) - 0.813(V-128) - 0.391(U-128)$$

$$R = 1.164(Y-16) + 1.596(V-128)$$

Fig. 38: Canal de información de color U

El resultado de esto es la imagen representada en la figura 39, una imagen que contiene toda la información de color y luminosidad.

Fig. 39: Los 3 canales combinados en una nueva imagen

3.5. DNR: Reducción de ruido dinámico

El sub-algoritmo de reducción de ruido dinámico (DNR), llamado *Mezclador* es solo válido para señales de video y procesa tanto la componente de color como la de luminosidad de la imagen, es por eso que previamente se han combinado las capas de iluminación y color en una imagen completa.

Este algoritmo está especializado en la reducción del ruido no coherente en la señal, sea del tipo aleatorio, ruido blanco o cualquier otro cuyo ruido no sea estático, o sea, que los píxeles afectados por él no estén en las mismas coordenadas de la imagen en dos imágenes consecutivas de un video.

Este algoritmo es de tipo mezclador temporal, razón por la cual solo tiene efecto en señales de video y no en imágenes estáticas.

Partiendo de la información original de la imagen actual, denominada *cimg* y la procesada de la imagen anterior llamada *pimg*, se comparan ambas y se seleccionan los píxeles similares, para determinar que sea similares se calcula la diferencia entre cada píxel de la imagen actual, llamado *cpix* y el de las mismas coordenadas en la imagen previa, llamado *ppix*; tenemos pues que:

$$dif = |(cpix - ppix)|$$

Si *dif*, que es la diferencia entre *cpix* y *ppix*, es menor a una constante *xdif* previamente establecida, entonces se procesa, de lo contrario *cpix* se deja con el valor actual. El procesado consiste en hacer una doble mezcla de los píxeles.

La primera mezcla se realiza con los valores de *cpix* y *ppix*, en esta mezcla el porcentaje del valor actual y el anterior se establece en relación al valor de *dif*, a mayor diferencia, mayor porcentaje tiene el valor actual y menor el anterior. De esta mezcla se obtiene *r*.

$$r = f(p, c, l) \text{ , con } l = \frac{w\,a}{j} \text{ y } w = |(c - p)|$$

Donde *r* es el resultado de la mezcla, p el valor previo, c el valor actual y l la relación de la mezcla que viene determinada por *w*, que es la diferencia entre *c* y *p*, la constante a con un valor recomendado >= 100, y *j* el valor máximo permitido de diferencia entre los valores *p* y *c*.

La función de mezcla *f*, está definida por: $f(s, d, l) = \frac{d\,l + s(a - l)}{a}$ con $0 \le l \le a$

Donde *s* es el primer valor de entrada, *d* el segundo, *l* la relación en la mezcla y la constante *a*.

La segunda mezcla establece una relación entre los valores *r* y *c* que viene determinada por la constante *m*, actuando ésta como un multiplicador del valor actual y que debe normalizarse en relación a la constante *a*.

$v = f(r, c, t(m)a)$, con $t(m) = m / tmax(m)$, donde *tmax* proporciona el máximo valor de *m*.

Esto hay que hacerlo por cada pareja de píxeles de *cimg* y *pimg*.

Este algoritmo tiene un modo de procesamiento llamado *único*, en el cual la información de la imagen actual (*cimg*) no se mezcla con la procesada de la anterior, si no que *pimg* es la imagen original anterior, esto es importante, pues permite establecer unos valores de relación altos para el

contenido previo sin que se generen halos como los que podemos observar en la figura 40.

Mezclador simple

El mezclador simple es un sub-algoritmo basado en el anterior, es una versión simplificada pero basada en el mismo principio, mucho más simple, con unos resultados menos precisos, pero que con valores bajos de diferencia genera buenos resultados, su principal baza es que es mucho más rápido y sencillo de implementar. En el algoritmo principal se ha usado la versión anteriormente descrita, pues da mejores resultados, pero hay casos donde el sub-algoritmo mezclador podría ser más indicado. Su funcionamiento consiste en que partiendo del valor máximo de diferencia *j*, los valores cuya diferencia sea superior a j/2 no se procesarán y por lo tanto v = p.

En los datos no filtrados, la relación del valor actual y el anterior es fijo, y *v* viene determinado por:

$$v = \frac{2c + p}{3}$$

Debido a que no se usa mezcla proporcional a la diferencia y que el mezclado tiene en cuenta el procesado anterior, no se pueden establecer valores de diferencia altos o se crearán halos en la imagen final, tal y como podemos observar en la figura 40.

Algo importante de estos sub-algoritmos, es que se conserva en gran medida la nitidez de la imagen cuando hay poco movimiento en una escena y que no se producen grandes alteraciones en el color, siempre y cuando establezcamos unos valores de procesamiento adecuados. Sin embargo, si es una escena con mucho movimiento en la imagen se pueden generar halos, para evitar esto, el algoritmo DNR tiene el modo de mezclado *único* y también se puede establecer un valor bajo de diferencia máxima.

Fig. 40: Halos producidos al mezclar dos imágenes de un video

3.6. Suavizado multicapa

Finalmente se realiza un proceso de suavizado sobre la imagen, este produce una imagen con menos grano y atenúa los cambios bruscos entre distintas intensidades de color y luminosidad, lo que deja una imagen más agradable a la vista. Por el contrario, la imagen pierde nitidez pues se eliminan las altas frecuencias, que es donde se encuentran los pequeños detalles, por lo que debemos ajustar muy bien el nivel con el que aplicamos el suavizado.

Aplicar este suavizado es importante en la mayoría de casos en los que se reduce ruido coherente, ya que en el proceso de reducción, el ruido residual que no hemos podido eliminar, queda muy contrastado respecto al resto del contenido de la imagen, y el suavizado ayuda a reducir este efecto.

En la figura 41 podemos ver un ejemplo de este proceso, a la izquierda tenemos la imagen original, la cual aún no ha sido procesada y contiene una señal de ruido coherente. En el centro de la figura tenemos la imagen una vez procesada, con el ruido coherente reducido, en donde podemos ver el ruido residual que no hemos podido eliminar. Y a la derecha de la figura tenemos la imagen después de aplicarle el suavizado.

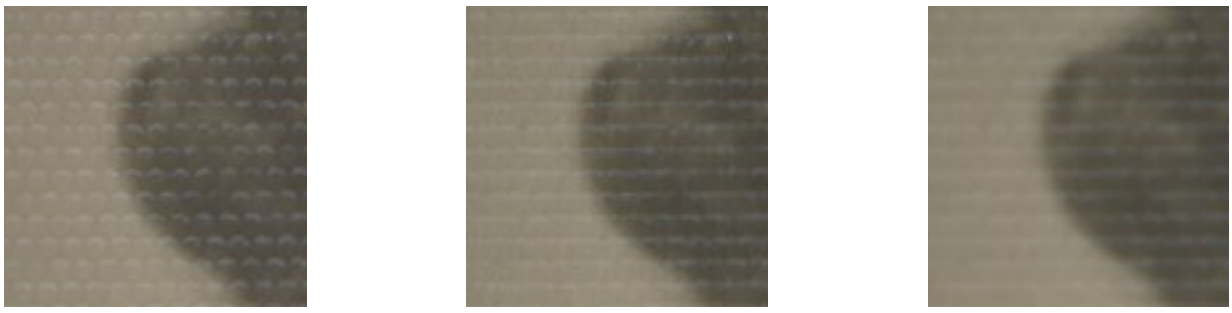

Fig. 41: Imagen original (izquierda), imagen con el ruido reducido (centro), imagen suavizada (derecha)

Para mostrar el funcionamiento de un filtro de suavizado, en la figura 42 podemos ver el resultado de este filtro con distintos radios, se usan 3 radios distintos, a la izquierda tenemos la imagen original, seguidamente un suavizado con un radio de un píxel, después de 4 píxels y finalmente con un radio de 16 píxeles.

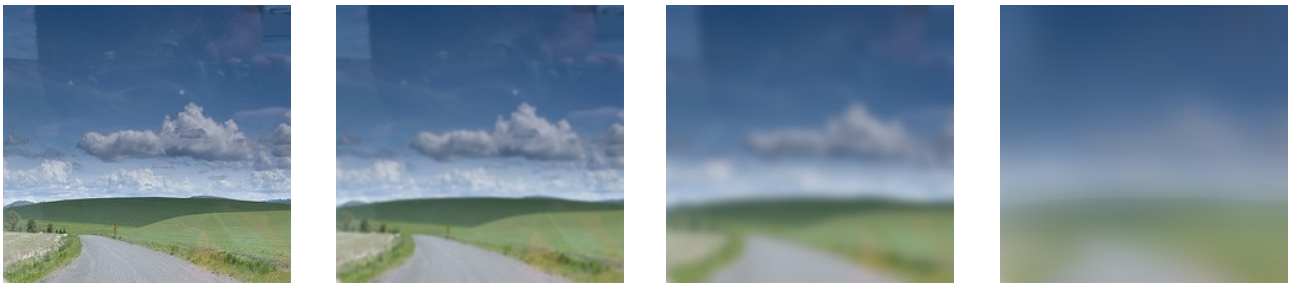

Fig. 42: Original *Radio 1px* *Radio 4px* *Radio 16px*

Como se puede ver, para el sub-algoritmo, el uso de radios superiores a 1 no tiene utilidad, pues dejaría la imagen muy borrosa y eliminaríamos no solo el ruido de la imagen, si no también casi todo su contenido. Ya que un filtro de suavizado elimina la información de alta frecuencia, contra mayor radio pongamos, se comienza a filtrar desde frecuencias más bajas en adelante.

Por lo tanto establecemos el radio a un valor fijo de 1 píxel y posteriormente el resultado se va a mezclar con la imagen original, pudiendo especificar la relación de la mezcla.

La función de mezcla *f*, está definida por: $f(s,d,l)=\frac{d\,l+s(a-l)}{a}$ con $0\leq l\leq a$

Donde *s* es el valor original, *d* el valor habiendo aplicado el suavizado, *l* la relación en la mezcla y *a* una constante de valor recomendado >= 100, la cual nos dará el número de pasos intermedios que se pueden obtener.

Con esto conseguimos obtener suavizados aún más sutiles, donde tenemos gran precisión de la intensidad con la que aplicar el filtro, lo cual nos permite no eliminar más información de la necesaria. Si *l* lo establecemos al valor máximo determinado por *a*, entonces obtendremos el resultado visto en la figura 42 con un radio de 1 píxel. Contra más bajo pongamos el valor de *l*, más nos aproximaremos a la imagen original.

Después de aplicar el suavizado ya tenemos el resultado final.

3.7. Resultado final

En las figuras 43 y 44, tenemos respectivamente las imágenes de entrada a nuestro algoritmo de reducción de ruido (la imagen original) y la imagen resultante después de aplicarlo.

Podemos ver que el ruido coherente que sale representado por líneas verticales a lo largo de toda la imagen ha sido reducido.

También podemos ver que en la imagen original cada zona que tiene un color distinto, posee un tono menos homogéneo que en la imagen procesada, ya que tiene pequeñas distorsiones de color; esto se puede observar sobre todo en las zonas blancas.

Fig. 43: Imagen original

Fig. 44: Imagen procesada. El ruido ha sido reducido y tenemos una imagen más suave tanto en color como luminosidad

► 4. Implementación y resultados

El algoritmo se ha implementado bajo la interfaz de un plugin de la aplicación de edición de video VirtualDub (www.virtualdub.org). Se ha realizado así porque la interfaz de programación que proporciona esta aplicación tiene todo lo que necesitamos para nuestro algoritmo, soporta diversos modelos de color, permite la carga de imágenes y soporta muchos formatos de video. Además es una aplicación gratuita y de código abierto.

Todo el código ha sido desarrollado en C++ con el compilador Visual C++ 2010.

Se ha usado la librería FFTW v2 para los algoritmos de transformada de Fourier y transformada inversa de Fourier, pues es la que mejores resultados ha dado y también la más rápida.

El tiempo de implementación ha sido de casi 4 meses; previamente hubo una etapa de investigación y pruebas de algo más de 3 meses y finalmente mes y medio para la documentación.

La máquina de pruebas tiene las siguientes especificaciones:

Procesador:	Intel i7 920 D0 @ 3.2 Ghz	Memoria:	18 GB DDR3 @ 1666 MHz
Gráfica:	ATI HD3870 512 MB GDDR5	Disco duro:	1TB 7200 rpm SATA2
Sistema:	Windows 7 64-bit	VirtualDub:	v1.9.9 32-bit

La configuración del algoritmo viene expuesta en el siguiente formato, el valor por defecto para cada parámetro viene indicado entre paréntesis mediante el indicador def.:

– cur_u, cur_v: Radio de suavizado en las capas de color U (2) y V (2) en el sub-algoritmo CNR
– dnr_lev, dnr_dif: Valor de *m (def.* 5) y de *j* (def. 255) del sub-algoritmo DNR.
– safe_center, safe_ext: Grosor de las zonas seguras del centro (def. 80) y la cruz (def. 5)
– dmask_level, dmask_dif, dmask_value: Nivel mínimo (*minl*) (def. 0), diferencia (*difl*) y valor de la constante de multiplicación (*cnt*) del sub-algoritmo de máscara dinámica.
– smask_int, smask_dim, smask_dif: Intensidad (*int*) (def. 0), diámetro (*rd*2*) y nivel del umbral (*dif*) del sub-algoritmo de máscara de desenfoque.
– mask_dif: Indica si está activo el sub-algoritmo de máscara diferencial (def. desactivado), aunque en las pruebas por defecto se establece como desactivado es el sub-algoritmo por defecto para el grupo PNR.
– int: Intensidad general (*gint*) del algoritmo de inversión (def. 255).
– soft: Valor del sub-algoritmo de suavizado, rango 0 a 10 (def. 5).

A continuación se exponen algunos ejemplos de los resultados obtenidos con este algoritmo, así como los parámetros usados para conseguirlos.

Todos los valores no indicados en la configuración de las constantes y parámetros, toman su valor por defecto si este se ha indicado, el resto toman valor 0 o desactivado. El tipo usado para la representación de un píxel en la imagen es de 8 bits por canal. El DNR esta en modo *único* y *diferencial* salvo que se indique lo contrario.

/ Ejemplo 1: PNR

Fig. 45: Imagen original

Fig. 46: Imagen procesada (config. 1)

Fig. 47: Imagen procesada (config.2)

Configuración del algoritmo:

- *Configuración 1*: cnr_u = 2, safe_center = 64, smask_int = 250, smask_dim = 43, smask_dif = 16, int = 205
- *Configuración 2*: cnr_u = 2, safe_center = 64, dmask_level = 130, dmask_value = 42, int = 205
- *Configuración 3*: cnr_u = 2, safe_center = 40, mask_dif = activa, int = 381

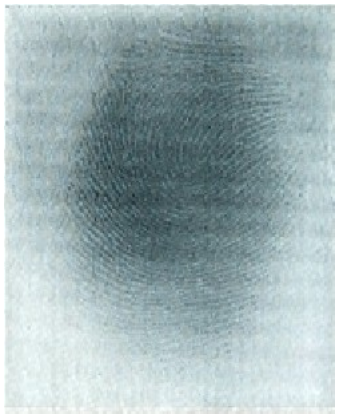

Fig. 48: Imagen procesada (config. 3)

En este ejemplo, tenemos una imagen estática de entrada (figura 45), con una señal de ruido coherente muy pronunciada que apenas deja ver el contenido real de la imagen; como podemos ver el algoritmo logra eliminar casi en su totalidad el ruido coherente.

Se ha procesado la imagen con tres configuraciones del algoritmo:

En la configuración 1 se ha usado el sub-algoritmo de selección mediante máscara de desenfoque (figura 46), mientras que en la configuración 2 se ha usado el sub-algoritmo de selección por máscara dinámica (figura 47); por último, en la configuración 3 se ha usado el sub-algoritmo de máscara dinámica (figura 48).

Pueden verse claramente las diferencias entre los resultados; mediante la configuración 1 se ha conseguido una imagen más nítida, con los contornos mejor definidos y más contraste. Con la configuración 2 en cambio se logra reducir el ruido coherente en mayor medida. La configuración 3 consigue unos resultados intermedios entre las otras dos, da una imagen nítida pero no elimina completamente el ruido.

/ Ejemplo 2: DNR

El sub-algoritmo DNR, está especializado en la eliminación de ruido dinámico y en los defectos de las imágenes de video producidos por la alta compresión de algunos métodos de compresión como el MJPEG, MPEG y sus variantes [7,15].

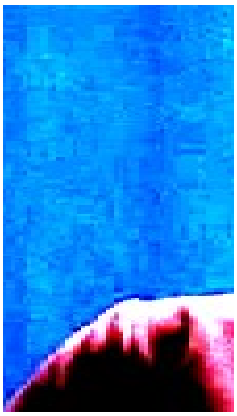

Fig. 50: Detalle de imagen original

Fig. 49: Imagen original

En el siguiente ejemplo podemos ver el resultado del sub-algoritmo DNR en una señal de video con macrobloques producidos por la compresión MPEG y una cantidad ligera de ruido dinámico. En la figura 49 podemos ver la imagen original antes de ser procesada; así mismo en la figura 50, tenemos una ampliación a la que se ha aumentado el contraste, que nos servirá para ver en detalle el ruido y defectos de la compresión.

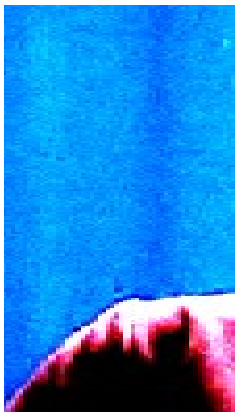

Fig. 52: Detalle de imagen procesada

Fig. 51: Imagen procesada

Como podemos observar en la figura 51 y ver con más detalle en la figura 52, la imagen procesada es mucho más suave, con un menor nivel de ruido y los defectos de la compresión se han reducido mucho.

Los parámetros usados han sido: dnr_lev = 10 y dnr_dif = 255, con el resto de grupos de reducción de ruido desactivados.

El resto de parámetros están a su valor por defecto.

/ Ejemplo 3: CNR

En el siguiente ejemplo podemos ver los resultados obtenidos por el algoritmo, a partir de la imagen con un alto nivel de ruido cromático representada en la figura 53.

Se han establecido dos configuraciones de prueba, la primera con unos valores óptimos, usando los siguientes valores en los parámetros: cnr_v = 9 y cnr_u = 7, soft = 0. Además el único bloque activo es el CNR. El resultado obtenido de esta configuración podemos verlo en la figura 54.

Podemos ver que el ruido cromático ha sido reducido, cada zona de color tiene un tono más homogéneo y se mantiene la nitidez de la imagen. También podemos observar que la saturación de los colores de la imagen ha disminuido, por lo que se deben ajustar los valores de reducción para no eliminar también los colores que deseamos mantener.

Como ejemplo de resultado de una reducción excesiva de ruido, tenemos la segunda configuración del ejemplo, en este caso hemos establecido unos parámetros iguales a los de la configuración anterior pero con unos valores superiores en la reducción de ruido cromático, los parámetros son: cnr_v = 19, y cnr_u = 20. El resultado obtenido es el de la figura 55, en la cual podemos observar como además de reducir en gran medida el ruido cromático, también se ha reducido de forma excesiva el color de la imagen.

Fig. 53: Imagen original

Fig. 54: Imagen resultante con parámetros óptimos:
cnr_v = *9,* cnr_u = *7*

Fig. 55: Imagen resultante con:
cnr_v = *19,* cnr_u = *20*

► 5. Conclusiones

El algoritmo desarrollado, consigue reducir de forma satisfactoria la mayoría de tipos de ruido típicos que se pueden encontrar en las imágenes estáticas y señales de video.

Se han conseguido diseñar unos sub-algoritmos para procesar correctamente la mayoría de casos sin precisar grandes ajustes manuales. También se tiene la posibilidad de usar otros sub-algoritmos que permiten un mayor ajuste, para ser usados en casos concretos donde resulte difícil la eliminación de ruido, principalmente en el tipo de ruido coherente. Para este tipo de ruido, el sub-algoritmo máscara diferencial consigue unos buenos resultados en la mayoría de ocasiones, siendo éste prácticamente automático, al no necesitar la configuración de ningún parámetro salvo la intensidad con la que es aplicado.

En base a las pruebas realizadas y como se puede observar en los ejemplos, los resultados para todos los tipos de ruido para los que ha sido diseñado son satisfactorias, pues consiguen el objetivo propuesto.

El segundo objetivo también se ha conseguido, se quería encontrar una configuración que fuese adecuada para la mayoría de casos prácticos y que el ajuste manual de los parámetros fuese mínimo. En los tres grupos de sub-algoritmos de reducción de ruido, se ha conseguido crear una configuración base que consigue este objetivo; si bien los grupos CNR y DNR son los que menos ajuste manual requieren y en la mayoría de ocasiones éste es innecesario; el grupo PNR es más complejo, pues el ruido coherente puede tener muchas más variaciones; aun así el sub-algoritmo máscara dinámica, que es el de uso por defecto, da unos buenos resultados en la mayoría de casos.

El algoritmo ha sido diseñado para obtener los mejores resultados posibles, pero la velocidad también ha sido una de las prioridades que se han tenido en cuenta, siempre y cuando esta no interfiera con la calidad del resultado. El diseño del algoritmo aquí detallado es bastante óptimo y puede ser implementado de forma que consuma unos recursos moderados, además la velocidad también depende de la configuración elegida y se han proporcionado algunos sub-algoritmos alternativos para implementaciones en las que impere la velocidad, como el del mezclador simple. Aunque estos sub-algoritmos alternativos son más sencillos y no proporcionan la misma calidad que los usados en el algoritmo principal, dan buenos resultados y en casos concretos pueden ser más adecuados para determinadas implementaciones. De este modo conseguimos todos los objetivos propuestos inicialmente.

► 6. Bibliografía

[1] Mari Mutt, J.A. (2003) Tratamiento digital de señales

[2] Rodríguez, Hugo. (2005) Imagen Digital Conceptos básicos

[3] Dan Margulis. (1998) Sharpening With a Stiletto.

[4] Alonso Fernández, Daniel. (2010) Análisis de técnicas de fusión de imágenes para su uso en fotografía digital.

[5] M.Sc Jimy Alexander Cortés Osorio. (1997) Técnicas alternativas para la conversión de imágenes a color a escala de grises en el tratamiento digital de imágenes. p. 207-210

[6] JL Contreras. Sistemas de coordenadas y vectores

[7] Barry G. Haskell. (1997) Digital video: An introduction to MPEG-2

[8] Kodak Digital GEM
http://www.kodak.com/global/en/professional/products/software/imgEnhancePlugIns/gem.jhtml?pq-path=13941

[9] Neat Image
http://www.neatimage.com/

[10] Clear ID
http://www.oceansystems.com/dtective/clearid/

[11] Fujifilm Grid Pattern Removal (GPR)
http://www.fujifilm.com/image_intelligence/medical_imaging/grid_pattern_removal/

[12] A Pizurica, V Zlokolica. (2003) Noise reduction in video sequences using wavelet-domain and temporal filtering

[13] Wampler, E. J. (1992) FFT removal of pattern noise in CCD images. p. 82-84

[14] WB Pennebaker. (1993) JPEG still image data compression standard

[15] Marpe, D., Wiegand, T., Sullivan, G.J. (2006) The H.264/MPEG4 advanced video coding standard and its applications